JS高级

Day01

课上练习

1.4 类的定义

```
// 练习1：定义人类Person，属性有name，sex，方法有run(方法里只要打印一句话即可)
class Person{
    constructor (theName, theSex) {
        this.name = theName;
        this.sex = theSex;
    }
    run () {
        console.log("人类会跑");
    }
    say () {
        console.log(this.name + "会说话");
    }
}
// 练习2：用Person实例化2个对象，并且传入相应的名字和性别，打印实例化的2个对象观察
var per1 = new Person("老黄", 18);
var per2 = new Person("老刘", 15);
console.log(per1);
console.log(per2);

// 练习3：在Person类里新增方法，say(方法里打印调用者名字)例如：小黄会说话，然后用上一个练习的实例对象，分别调用run方法执行，查看打印结果
per1.say();
per2.say();

// 练习4：定义学生类Student，属性有name，sex，hobby，方法有play
class Student{
    constructor (theName, theSex, theHobby) {
        this.name = theName;
        this.sex = theSex;
        this.hobby = theHobby;
    }
    play() {
        console.log(`${this.name}在${this.hobby}`);
    }
}

// 练习5：用Student类，实例化1个对象，调用play方法，play方法里return一个字符串，字符串格式为谁在干什么爱好，例如：小明在玩篮球(此时name的值是小明，hobby的值是篮球)，外部接受return的字符串并打印
var stu = new Student("小传", "女", "写代码");
stu.play();
```

```
// 练习6：定义工具类Tool，实现2个方法，一个求3个数的和，一个求3个数里最大值，并且都要return返回结果，再调用处打印结果即可
class Tool {
    getSum(a, b, c) {
        return a + b + c;
    }
    getMax(a, b, c) {
        return Math.max(a, b, c);
    }
}
var toolObj = new Tool();
var result = toolObj.getSum(10, 20, 30);
console.log(result);
var theMax = toolObj.getMax(100, 200, 50);
console.log(theMax);
```

2.3 继承的练习

```
// 练习：
/*
Student类继承自Person类
Person里属性：name和sex
Person里方法：run - 打印一个字符串即可
say - 打印当前调用者名字
Student里属性：hobby
Student里方法：play - 返回拼接的字符串：调用者名字和爱好

实例化子类对象，传入3个实际参数的值
提示，子类里的super()应该把收到的实参值传递给父constructor函数，把值绑定到实例对象的name和sex属性上

最后调用run和say观察打印结果
调用play方法拿到返回值，再new下面打印返回值(不要再play函数里打印)
*/
```

js答案:

```
class Person {
    constructor(theName, theSex) {
        this.name = theName;
        this.sex = theSex;
    }
    run() {
        console.log("我是run方法");
    }
    say() {
        console.log(this.name + "会说话");
    }
}

class Student extends Person {

    constructor(theName, theSex, theHobby) {
```

```
        super(theName, theSex);
        this.hobby = theHobby;
    }
    play() {
        return `${this.name} 在玩 ${this.hobby}`;
    }
}


var stu = new Student("小花", "女", "跳皮筋");
var result = stu.play();
console.log(result); // 小花 在玩 跳皮筋
stu.run();
stu.say();
```

案例

5. class封装tab栏

了解插件如何封装和调用以及复用 - 模拟new Swiper的实现过程

先演示如何使用定义好的插件

5.0 tab栏_标签和样式

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>案例_封装tab栏_标签和样式准备</title>
        <style>
            .box {
                border: 1px solid #000;
                box-sizing: border-box;
            }
            .box * {
                box-sizing: border-box;
            }
            .box .header {
                width: 100%;
                height: 50px;
                display: flex;
                border-bottom: 1px solid #222;

            }
```

```
            .box .header span {
                transition: all 0.3s;
                background-color: #fff;
                line-height: 50px;
                text-align: center;
                cursor: pointer;
                font-size: 16px;
                border-right: 1px solid #000;
                flex: 1;
            }

            .box .header span.ac,
            .box .header span:hover {
                background-color: #ccc;
            }

            .box .main {
                width: 100%;
            }

            .box .main div {
                width: 100%;
                text-align: center;
                font-size: 60px;
                font-weight: 800;
                display: none;
            }
            .box .main div.active {
                display: block;
            }
        </style>
    </head>

    <body>
        <!-- 标签栏容器 -->
        <div class="box">
            <!-- 导航 -->
            <div class="header">
                <span class="ac">tab1</span>
                <span>tab2</span>
                <span>tab3</span>
            </div>
            <!-- 内容 -->
            <div class="main">
                <div class="active">内容1</div>
                <div>内容2</div>
                <div>内容3</div>
            </div>
        </div>
    </body>

</html>
```

5.1 tab栏_class类定义

我们想要多个tab栏, 所以需要tab栏模板, 封装一个类, 模拟swiper的使用

```
// 1. 定义类
class Tab {
    constructor(query, configObj) {
        // 3.1 接收容器和配置
        this.box = document.querySelector(query);
        this.header = this.box.querySelector(".header");
        this.main = this.box.querySelector(".main");
        // 导航数据和正文数据
        this.tabTitleArr = configObj.tabTitleArr;
        this.contentArr = configObj.contentArr;
        // 3.2 调用初始化标签方法
        this.init();
    }
    init() {
        // 3.3 模板字符串生成标签
        var spanStr = ``;
        var divStr = ``;
        this.tabTitleArr.forEach(function (tabStr, ind) {
            spanStr += `<span class="${ind == 0 ? 'ac' : ''}">${tabStr}</span>`;
        })
        this.contentArr.forEach(function (val, index) {
            divStr += `<div class="${index == 0 ? 'active' : ''}">${val}</div>`
        })
        // 3.4 把标签字符串插入到指定位置
        this.header.innerHTML = spanStr;
        this.main.innerHTML = divStr;
    }
}

// 2. 模拟swiper使用, new 类, 传入标签容器选择器, 传入配置对象
new Tab("#box", {
    tabTitleArr: ["tab1", "tab2", "tab3"],
    contentArr: ["内容1", "内容2", "<span style='color: red;'>内容3</span>"]
})
```

5.2 tab栏_点击切换

```
// 1. 定义类
var that;
class Tab {
    constructor(query, configObj) {
        that = this;
        // 3.1 接收容器和配置
        this.box = document.querySelector(query);
        this.header = this.box.querySelector(".header");
        this.main = this.box.querySelector(".main");
        // 导航数据和正文数据

        this.tabTitleArr = configObj.tabTitleArr;
```

```
        this.contentArr = configObj.contentArr;
        // 4.1 声明属性，保存新建的span导航和div内容标签
        this.headerSpanArr = [];
        this.mainDivArr = [];
        // 3.2 调用初始化标签方法
        this.init();
    }
    init() {
        // 3.3 模板字符串生成标签
        var spanStr = ``;
        var divStr = ``;
        this.tabTitleArr.forEach(function (tabStr, ind) {
            // 4.4 给span - 绑定索引
            spanStr += `<span index="${ind}" class="${ind == 0 ? 'ac' : ''}">${tabStr}</span>`;
        })
        this.contentArr.forEach(function (val, index) {
            divStr += `<div class="${index == 0 ? 'active' : ''}">${val}</div>`
        })
        // 3.4 把标签字符串插入到指定位置
        this.header.innerHTML = spanStr;
        this.main.innerHTML = divStr;

        // 4.2 获取所有的导航span和内容div标签
        this.headerSpanArr = Array.from(this.header.querySelectorAll("span"));
        this.mainDivArr = Array.from(this.main.querySelectorAll("div"));
        // 4.3 给this.header绑定事件 - 事件委托
        this.header.onclick = this.toggleTab;
    }
    toggleTab(ev) {
        // 4.5 获取索引
        var index = ev.target.getAttribute("index");
        // 4.6 让当前标签高亮
        that.headerSpanArr.forEach(function (span, ind) {
            span.className = "";
            that.mainDivArr[ind].className = "";
        })
        // 4.7 让当前高亮
        ev.target.className = "ac";
        that.mainDivArr[index].className = "active";
    }
}

// 2. 模拟swiper使用，new 类，传入标签容器选择器，传入配置对象
new Tab("#box", {
    tabTitleArr: ["tab1", "tab2", "tab3"],
    contentArr: ["内容1", "内容2", "<span style='color: red;'>内容3</span>"]
})
```

5.3 tab栏_编写使用文档

好了, 功能实现后, 可以自己写一个文档, 告诉别人怎么去使用了

· index.css - 样式文件

· index.js - 主要功能

· 使用插件者, 引入index.css和index.js, 然后准备标签结构, 只需要写一个new Tab()传入相关参数就有了tab栏切换的案例

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>我要一个tab栏_带切换的</title>
        <link rel="stylesheet" href="./css/index.css">
    </head>
    <body>
        <div class="box" id="box">
            <div class="header">
            </div>
            <div class="main">
            </div>
        </div>
        <script src="./js/index.js"></script>
        <script>
            new Tab("#box", {
                tabTitleArr: ["体育", "军事", "娱乐"],
                contentArr: ["我是体育页面", "军事内容好啊", "这里可以放图片啊或者完整的标签结构啊，但是注意用模板字符串哦，假设我是娱乐新闻"]
            })
        </script>
    </body>
</html>
```

作业

1. 使用class关键字, 自定义时间类MyDate, 继承自系统的时间类Date -

(要求1): 在自定义类MyDate里新增1个方法叫formatDate, 返回格式化的日期YYYY/MM/DD HH:mm:cc (要求, 在方法内, 请使用this来获取年月日, 思考this是什么? 不要再重新new Date())

(要求2): 实例化自定义类的实例对象, 调用getFullYear()查看返回值, 调用自己定义的formatDate()查看返回值

答案:

```
<script>
    // 使用class关键字，自定义时间类MyDate，继承自系统的时间类Date -

    // (要求1): 在自定义类MyDate里新增1个方法叫formatDate，返回格式化的日期YYYY/MM/DD HH:mm:cc (要求，在方法内，请使用this来获取年月日，思考this是什么? 不要再重新new Date())

    // (要求2): 实例化自定义类的实例对象，调用getFullYear()查看返回值，调用自己定义的formatDate()查看返回值

    class MyDate extends Date {
        formatDate() {
```

```
            var year = this.getFullYear();
            var month = this.getMonth() + 1;
            var day = this.getDate();
            var hours = this.getHours();
            var minutes = this.getMinutes();
            var seconds = this.getSeconds();

            var arr = [year, month, day, hours, minutes, seconds].map(function (val) {
                return val < 10 ? '0' + val : val;
            });
            return arr.slice(0, 3).join("/") + " " + arr.slice(3).join(":")
        }
    }
var a = new MyDate();
console.log(a.formatDate());
</script>
```

1. 把之前JS基础/webAPI写过的工具方法, 尝试封装到工具类中 (写多少交多少)

例如:

```
class XDTool {
  getRandomColor() { // 返回随机数颜色rgb()字符串

  }

  getMax(a, b) { // 求2个数最大值

  }

  getArraySum(arr) { // 传入数组，在这里用forEach遍历求和返回

  }
}

// 2个以上方法都要使用的变量值，可以在constructor里绑定在this的属性上(函数内this.xxx调用)

// 而只有当前方法自己使用的值，传入到形参中即可(例如a，b求2个数最大值)
```

答案:

```
class XDTool {
    getRandomColor() { // 返回随机数颜色rgb()字符串
        var r = Math.floor(Math.random() * 256);
        var g = Math.floor(Math.random() * 256);
        var b = Math.floor(Math.random() * 256);
        return `rgb(${r}, ${g}, ${b})`;
    }
    getMax(a, b) { // 求2个数最大值
        return Math.max(a, b);
    }
    getArraySum(arr) { // 传入数组，在这里用forEach遍历求和返回
```

```
        var sum = 0;
        arr.forEach(function(val, index){
            sum = sum + val;
        })
        return sum;
    }
}
var d = new XDTool();
console.log(d.getArraySum([1, 2, 3]));
// 2个以上方法都要使用的变量值，可以在constructor里绑定在this的属性上(函数内this.xxx调用)
// 而只有当前方法自己使用的值，传入到形参中即可(例如a，b求2个数最大值)
```

Day02

课上练习

6.2 构造函数练习

练习1. 请定义一个构造函数Person, 有姓名, 身高, 体重, 家乡, sing方法. 方法中打印一句话"学习唱歌", 并实例化2个对象打印即可. (体会下如何创造模板)

```
// 练习1. 请定义一个构造函数Person，有姓名，身高，体重，家乡，sing方法. 方法中打印一句话"学习唱歌"，并实例化2个对象打印即可. (体会下如何创造模板)
function Person(uName, height, weight, home) {
    this.name = uName;
    this.height = height;
    this.weight = weight;
    this.home = home;
    this.sing = function(){
        console.log("学习唱歌");
    }
}

var per1 = new Person("小传", 189, 180, "北京");
console.log(per1);
var per2 = new Person("小黑", 210, 120, "顺义");
console.log(per2);
```

7.8 工具构造函数

```
练习1：定义ChuanTool构造函数，属性接受2个数字(numA，numB)，定义2个方法(getMax，randomNum)，getMax方法返回numA和numB其中比较大的值，另一个方法返回一个随机数(范围就是numA到numB，假设numA是小于numB的)
练习2：实例对象调用getMax，和randomNum方法，再使用toString()方法 - 画出查找的方法机制规则(原型链，包括每个部分详细的信息，对象里的key和value是什么都画)
练习3：给系统内置的构造函数Array扩展一个求最大值的方法
```

答案:

```
/*
```

```
        练习1：定义ChuanTool构造函数，属性接受2个数字(numA，numB)，定义2个方法(getMax，randomNum)，getMax方法返回numA和numB其中比较大的值，另一个方法返回一个随机数(范围就是numA到numB，假设numA是小于numB的)
        练习2：实例对象调用getMax，和randomNum方法，再使用toString()方法 - 画出查找的方法机制规则(原型链，包括每个部分详细的信息，对象里的key和value是什么都画)
        练习3：给系统内置的构造函数Array扩展一个求最大值的方法
    */
function ChuanTool(numA, numB){
    this.numA = numA;
    this.numB = numB;
}
ChuanTool.prototype.getMax = function(){
    return Math.max(this.numA, this.numB);
}
ChuanTool.prototype.randomNum = function(){
    return Math.floor(Math.random() * (this.numB - this.numA + 1) + this.numA);
}
var toolObj = new ChuanTool(5, 10);
console.log(toolObj.getMax());
console.log(toolObj.randomNum());
console.log(toolObj.toString());

// 练习3:
Array.prototype.getMax = function(){
    return Math.max(...this);
}
var arr = [5, 10, 15, 13, 2, 1];
console.log(arr.getMax());
```

8.5 继承 - 构造函数和原型

```
// 父类Person: 属性(uname, age, sex, hobby, address), 方法(eat / run)
// 子类Student: 属性(className, classNum), 方法(study)
// 子类Boy: 属性(gfName), 方法(play)
// 要求: Boy继承于Student, Student继承于Person (请使用构造函数+组合继承方式实现)

// 以后实际开发时候，继承一层就够了 (继承就是把公共的东西提升到父类里即可)
```

答案:

```
// 父类Person: 属性(name, age), 方法(eat / run)
// 子类Student: 属性(className), 方法(study)
// 子类Boy: 属性(gfName), 方法(play)
// 要求: Boy继承于Student, Student继承于Person (请使用构造函数+组合继承方式实现), 实例化Boy对象，传入所有属性，打印查看效果，调用父类方法查看效果

function Person(name, age) {
    this.name = name;
    this.age = age;
}

Person.prototype.eat = function(){
```

```
    console.log("人类会吃饭");
}
Person.prototype.run = function(){
    console.log(`${this.name} 会跑`);
}

function Student(name, age, className){
    Person.call(this, name, age);
    this.className = className;
}
Student.prototype = Object.assign({}, Person.prototype);
Student.prototype.study = function(){
    console.log("学生会学习");
}

function Boy(name, age, className, gfName){
    Student.call(this, name, age, className);
    this.gfName = gfName;
}
Boy.prototype = Object.assign({}, Student.prototype);
Boy.prototype.play = function(){
    console.log("男孩会玩");
}

var boy = new Boy("小明", 18, "顺义1期", "代码");
console.log(boy);
boy.eat();
boy.run();
boy.study();
boy.play();
```

案例

暂无

作业

答案:

```
// 需求1：尝试给现在系统的Array构造函数添加一个冒泡排序的函数，确保下面代码正确
// 提示：对象调用函数，本身没有会去哪里找？所以在调用之前可以在Array函数，的什么位置上扩展添加方法呢？
/*这里应该补充什么代码呢?*/
Array.prototype.bubbleSort = function () {
    for (var i = 0; i < this.length - 1; i++) {
        for (var j = 0; j < this.length - i - 1; j++) {
            if (this[j] > this[j + 1]) {
                var temp = this[j];
                this[j] = this[j + 1];
                this[j + 1] = temp;
            }
        }

    }
```

```
    return this;
}

/**********************/
var arr = [5, 2, 1, 10, 3, 4];
console.log(arr.bubbleSort());

// 需求2: 给系统Date构造函数, 添加一个toMyString函数 - 调用函数, 得到今天日期YYYY年MM月DD日 HH:mm:ss格式的时间
Date.prototype.toMyString = function () {
    var year = this.getFullYear();
    var month = this.getMonth() + 1;
    var day = this.getDate();
    var hours = this.getHours();
    var minutes = this.getMinutes();
    var seconds = this.getSeconds();

    var arr = [year, month, day, hours, minutes, seconds].map(function (val) {
        return val < 10 ? '0' + val : val;
    });
    return arr.slice(0, 3).join("/") + " " + arr.slice(3).join(":")
}
var d = new Date();
console.log(d.toMyString());
```

Day03

课上练习

9.7 闭包和递归练习

```
// 1. 闭包使用 - 给10个li绑定鼠标移入事件, 移入打印索引
// 2. 递归函数 - 用递归函数求5的阶乘 myFn(5)
```

正确答案:

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>课上练习</title>
    </head>

    <body>
        <!-- ul#myUL>li{我是$}*10 -->
        <ul id="myUL">
            <li>我是1</li>

            <li>我是2</li>
```

```
            <li>我是3</li>
            <li>我是4</li>
            <li>我是5</li>
            <li>我是6</li>
            <li>我是7</li>
            <li>我是8</li>
            <li>我是9</li>
            <li>我是10</li>
        </ul>
        <script>
            var a = 10;
            var b = 20;
            console.log(a + b);
            // 1. 自己实现一次数组的forEach循环方法
            // Array.prototype.myForEach = function(){//...补全代码，提示用普通for循环+回调函数}
            Array.prototype.myForEach = function(callbackFn){
                for (var i = 0; i < this.length; i++) {
                    callbackFn(this[i], i, this); // 每次循环，就回调函数执行一次，同时，把当前遍历的值回调传递回去
                }
            }
            var arr = [5, 6, 9, 2];
            arr.myForEach(function(value, index, array){
                console.log(value, index, array);
            }); // 系统内置的方法内部遍历的过程

            // 2. 闭包使用 - 给10个li绑定鼠标移入事件，移入打印索引
            var liList = document.querySelectorAll("#myUL>li");
            Array.from(liList).forEach(function(li, index){
                li.onmouseenter = function(){
                    console.log(index);
                }
            })

            // 3. 递归函数 - 用递归函数求5的阶乘 myFn(5)  (5 * 4 * 3 * 2 * 1)
            function myFn(n){
                if (n == 1) {
                    return 1;
                }
                return n * myFn(--n);
            }

            var result = myFn(5);
            console.log(result);
        </script>
    </body>

</html>
```

10.2 对象内存地址练习

```
// 请回答，下面判断表达式的结果
var obj = {
    a: 10,
    b: 20
}
var obj2 = {
    a: 10,
    b: 20
}
console.log(obj === obj2);
console.log(obj instanceof Object);
console.log(obj.__proto__.constructor === Object);
```

案例

11.10 - 案例 - 验证码发送

标签和样式准备:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>案例_验证码倒计时</title>
  <style>
    .form {
      width: 600px;
      margin: 100px auto;
    }

    .row {
      width: 100%;
      display: flex;
      border-top: 1px solid #0094ff;
      border-bottom: 1px solid #0094ff;
    }


    .cols-1 {
```

```
      flex: 1;
    }

    .cols-2 {
      flex: 2;
    }

    .cell {
      border-left: 1px solid #0094ff;
      padding: 10px 6px;
      line-height: 42px;
      text-align: center;
    }

    .cell:last-child {
      border-right: 1px solid #0094ff;
    }

    .text {
      width: 268px;
      height: 30px;
      padding: 4px;
    }

    .btn {
      height: 38px;
      width: 200px;
    }
  </style>
</head>

<body>
  <div class="form">
    <div class="row">
      <div class="cell cols-1">请输入手机号码</div>
      <div class="cell cols-2">
        <input type="text" class="text" id="phone">
      </div>
      <div class="cell cols-1">
        <input type="button" class="btn" value="获取验证码" id="getCode" disabled>
      </div>
    </div>
  </div>
</body>
<script>

</script>

</html>
```

正确的js代码:

```
// 需求：输入手机号，监测是否符合正则判断表达式，符合后才可以点击发送验证码，然后做倒计时效果
```

```
// 思路：监测手机号输入框的改变，影响发送验证码按钮的状态，给发送验证码按钮绑定点击事件，做验证码倒计时效果

var thePhone = document.getElementById("phone");
var getCodeBtn = document.getElementById("getCode");

thePhone.oninput = function(){
    if(/^1[345678]\d{9}$/.test(this.value)){
        getCodeBtn.disabled = false;
    } else {
        getCodeBtn.disabled = true;
    }
}

getCodeBtn.onclick = function () {
    getCodeBtn.disabled = true;
    thePhone.disabled = true;
    var i = 3;
    getCodeBtn.value = '验证码已发送' + i;

    var timer = window.setInterval(function () {
        i--;
        getCodeBtn.value = '验证码已发送' + i;
        if (i === 0) {
            window.clearInterval(timer);
            getCodeBtn.value = '获取验证码';
            thePhone.disabled = false;
            getCodeBtn.disabled = false;
        }
    }, 1000);
}
```

实际开发时 - 可以百度找到正则表达式 - 直接使用 (但是要学会读懂和修改)

笔试题

```
// 回答下面打印的结果是多少
function test(){
    var arr = [];
    for (var i = 0; i < 10; i++) {
        arr[i] = function(){
            console.log(i);
        }
    }
    return arr;
}
var myArr = test();
for (var j = 0; j < 10; j++) {
    myArr[j]();
}
// 提示：i变量虽然释放不掉，但是i可以被修改（注意执行的时机和顺序），而且作用域链不是在运行时决定的

// 解释：test只执行一次，var i 是test作用域下的一个变量（然后一直被for修改），最终i的值是10，所以下面调
```

用arr[i]后面函数体执行时，根据作用域链向test作用域下找到i访问，所以都是10

作业

```
// 1. 思考下面这行代码，请说出最后的结果
var arr = [5, 6, 7, 8];
arr.map(function(val){
    val = val + 1;
})
console.log(arr);

// 2. 思考下面这行代码，请说出最后的结果
var arr = [5, 6, 7, 8];
var brr = arr.map(function(val){
    val = val + 1;
    return val;
})
console.log(brr);

// 3. 请使用map循环，给对象里的每个人年龄+1，最后打印数组，回答为什么对象元素，就可以影响原数组里的对象？(不用deepCopy，因为就2层)
var brr = [{
    name: "小黑",
    age: 10
}, {
    name: "小王",
    age: 20
}];

// 4题. 在3题的基础上，通过map方法返回一个新的数组resultArr(里面对象数据和brr一样(但是年龄要+1)) - (注意不要影响brr的值) - 最后同时打印brr和resultArr
var resultArr = brr.map(function(val){

})

// 附加题-不要求必须做
// 斐波那契数列
// 1、1、2、3、5、8、13、21、34
// 请使用递归实现，返回第几位的斐波那契的值
// 效果：myFn(7) -> 返回13
// 提示：递归时，每次返回的应该是前2位的和

// 使用递归思路，找到id匹配的值，例如找到id=112的对象，并打印出来即可(return/不return都可以，如果想要接收出来了)
var data = [{
    id: 1,
    name: '家电',
    goods: [{
        id: 11,
        gname: '冰箱',
        goods: [{
            id: 111,
```

```
            gname: '海尔'
        }, {
            id: 112,
            gname: '美的'
        }, ]
    }, {
        id: 12,
        gname: '洗衣机'
    }]
}, {
    id: 2,
    name: '服饰'
}];
// 如果层数不确定怎么办，所以必须用递归实现
```

正确答案:

```
// 1. 思考下面这行代码，请说出最后的结果
var arr = [5, 6, 7, 8];
arr.map(function (val) {
    val = val + 1;
})
console.log(arr); // [5, 6, 7, 8]

// 2. 思考下面这行代码，请说出最后的结果
var arr = [5, 6, 7, 8];
var brr = arr.map(function (val) {
    val = val + 1;
    return val;
})
console.log(brr); // [6, 7, 8, 9]

// 3. 请使用map循环，给对象里的每个人年龄+1，最后打印数组，回答为什么对象元素，就可以影响原数组里的对象？(不用deepCopy，因为就2层)
var brr = [{
    name: "小黑",
    age: 10
}, {
    name: "小王",
    age: 20
}];
// brr.map(function (val) { // 引用类型在普通的赋值=过程是地址的复制，会互相影响
//     val['age'] = val['age'] + 1;
// })
// console.log(brr);

// 4题. 在3题的基础上，通过map方法返回一个新的数组resultArr(里面对象数据和brr一样(但是年龄要+1)) - (注意不要影响brr的值) - 最后同时打印brr和resultArr
var resultArr = brr.map(function (val) { // 内容复制过来就不会互相影响了
    var obj = {};
    for (var k in val) {

        obj[k] = val[k];
```

```
    }
    obj['age'] = obj['age'] + 1;
    return obj;
})
console.log(brr);
console.log(resultArr);

// 附加题-不要求必须做
// 斐波那契数列
// 1、1、2、3、5、8、13、21、34
// 请使用递归实现，返回第几位的斐波那契的值
// 效果：myFn(7) -> 返回13
// 提示：递归时，每次返回的应该是前2位的和
function myFn(n){ // 第一次调用n是4
    if (n == 1 || n == 2){ // n指的是位数，1就是第一位
        return 1; // 是值1
    }

    return myFn(n - 2) + myFn(n - 1);
}

console.log(myFn(1));
console.log(myFn(2));
console.log(myFn(3));
console.log(myFn(4));
//    console.log(myFn(5));
//    console.log(myFn(6));
//    console.log(myFn(7));

// 使用递归思路，找到id匹配的值，例如找到id=112的对象，并打印出来即可(return/不return都可以)
var data = [{
    id: 1,
    name: '家电',
    goods: [{
        id: 11,
        gname: '冰箱',
        goods: [{
            id: 111,
            gname: '海尔'
        }, {
            id: 112,
            gname: '美的'
        },]
    }, {
        id: 12,
        gname: '洗衣机'
    }]
}, {
    id: 2,
    name: '服饰'
}];
```

```
function find(data, id){
    var o;
    data.forEach(function(obj){
        if (obj.id === id) {
            console.log(obj);
            o = obj;
        } else if (obj.goods ) {
            o = find(obj.goods, id);
        }
    })
    return o;
}

var result = find(data, 112);
console.log(result);
```

Day04

课上练习

暂无

案例

12.4 案例 - 使用let实现循环绑定点击事件

例子: 循环遍历加监听, 使用let取代var是趋势

```
var liList = document.querySelectorAll("#myUL>li");
// 方式1. 自定义属性
// for (var i = 0; i < liList.length; i++) {
//     liList[i].index = i;
//     liList[i].onclick = function(){
//         console.log(this.index);
//     }
// }

// 方式2. 闭包
// for (var i = 0; i < liList.length; i++) {
//     (function (ind) {
//         liList[ind].onclick = function () {
//             console.log(ind);
//         }
//     })(i);
// }

// 方式3. let
for (let i = 0; i < liList.length; i++) {
   liList[i].onclick = function () {
       console.log(i);
```

```
    }
}
```

拆解过程, for循环每次i都在一个独立的块作用域内, JS引擎会记录上一次i的值是多少, 分配给你这次循环i的值

16.1 案例 - 全选和反选

/Day01/1-代码/4.3_案例_小多选框_影响全选.html

全选/全不选	菜名	商家	价格
	红烧肉	隆江猪脚饭	￥200
	香酥排骨	隆江猪脚饭	￥998
	北京烤鸭	隆江猪脚饭	￥88

```
// 需求：点击小多选框，都勾选时，全选框也勾选
// 思路：声明个变量，遍历每个小多选框，如有一个未选中，则变量直接保存false，赋予给全选框，否则全选框设置为true

// 1. 获取标签
var checkAll = document.getElementById("checkAll");
var ckList = document.querySelectorAll(".ck"); // 所有小多选框

// 2. 全选影响所有小的
checkAll.onclick = function(){
    var allChecked = this.checked;
    Array.from(ckList).map(function(el){
        el.checked = allChecked;
    })
}

// 3. 小影响多
var ckArr = Array.from(ckList);
ckArr.map(function(el){
    el.onclick = function(){
        var isAll = ckArr.every(function(el){return el.checked == true}); // 筛选是否有不符合条件的返回false
        checkAll.checked = isAll == false ? false : true;
    }
})
```

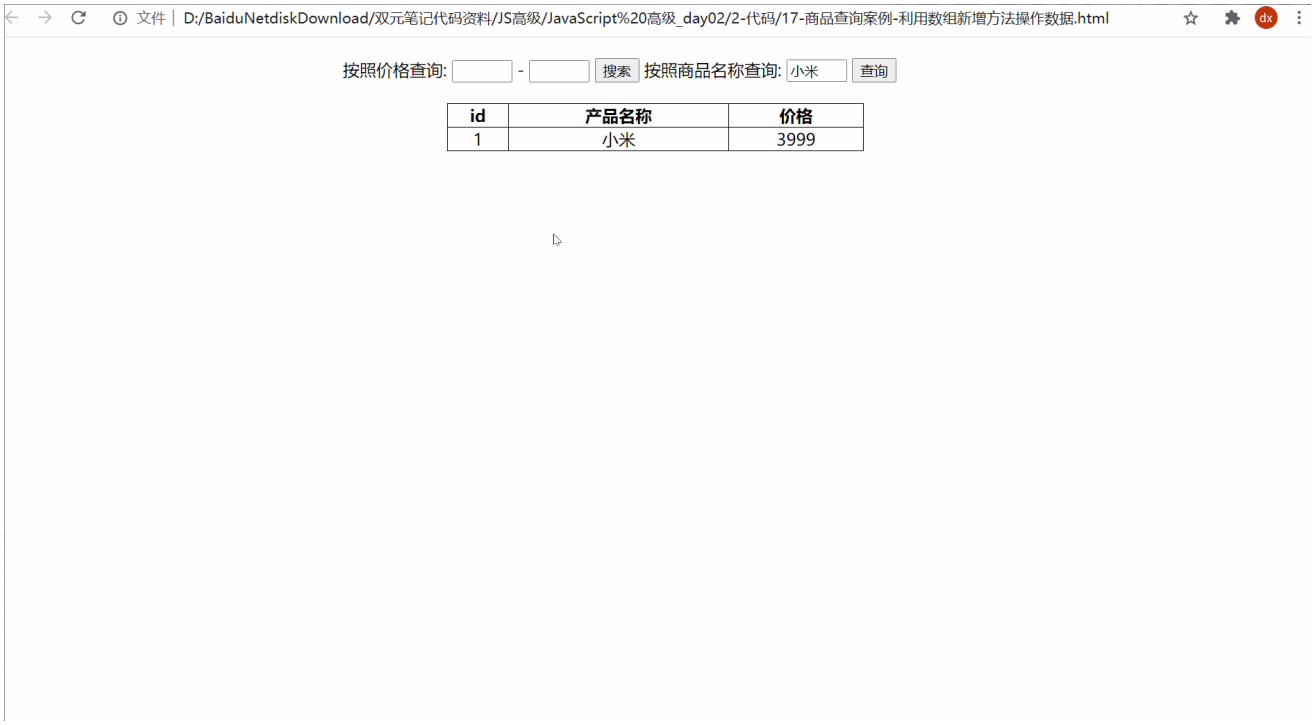
作业

请使用最新学的语法

- 模板字符串, 负责把JS里的数据, 铺设到页面表格中(模板字符串生成标签结构)
- ES5新增数组方法: 筛选符合条件的元素, 使用数组方法, 回调函数使用箭头函数

效果图

- 搜索价格范围, 显示符合条件的商品
- 名称搜索是全相等才显示对应商品
- 标签和JS数据暂无答案:



- 标签和JS数据

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Document</title>
        <style>
            table {
                width: 400px;
                border: 1px solid #000;
                border-collapse: collapse;
                margin: 0 auto;
            }

            td,
            th {
                border: 1px solid #000;

                text-align: center;
```

```
            }

            input {
                width: 50px;
            }

            .search {
                width: 600px;
                margin: 20px auto;
            }
        </style>
    </head>

    <body>
        <div class="search">
            按照价格查询: <input type="text" class="start"> - <input type="text" class="end">
<button class="search-price">搜索</button> 按照商品名称查询: <input type="text" class="product">
<button class="search-pro">查询</button>
        </div>
        <table>
            <thead>
                <tr>
                    <th>id</th>
                    <th>产品名称</th>
                    <th>价格</th>
                </tr>
            </thead>
            <tbody>


            </tbody>
        </table>
        <script>
            // 利用新增数组方法操作数据
            var data = [{
                id: 1,
                pname: '小米',
                price: 3999
            }, {
                id: 2,
                pname: 'oppo',
                price: 999
            }, {
                id: 3,
                pname: '荣耀',
                price: 1299
            }, {
                id: 4,
                pname: '华为',
                price: 1999
            }, ];
        </script>

    </body>
```

```
</html>
```

正确答案js代码

```
var tbody = document.querySelector("tbody");
var searchBtn = document.querySelector(".search-price");
var startInp = document.querySelector(".start");
var endInp = document.querySelector(".end");
// 1. 把数据铺设到页面上
function load(arr){
    // 2. 铺设页面 - JS基本功(必须张手就来) - 你在心中要把数据的结构和标签对应上
    // 对象 -> tr+td标签结构
    tbody.innerHTML = "";
    arr.forEach(function(obj){
        let {id, pname, price} = obj;
        var theTr = `<tr>
<td>${id}</td>
<td>${pname}</td>
<td>${price}</td>
</tr>`;
        tbody.innerHTML += theTr;
    })
}
// 网页打开的一瞬间 - 所有的数据data数组里的一切要加载一遍
load(data)


// 3. 价格搜索的按钮 - 点击事件
searchBtn.onclick = function(){
    var sPrice = startInp.value;
    var ePrice = endInp.value;
    // 4. 过滤data数组里在这个价格区间的产品对象
    var newArr = data.filter(function(obj){
        return obj.price >= sPrice && obj.price <= ePrice;
    });
    load(newArr);
}

// 5. 思路: 跟上面一样, 就是return 换成 obj.name == 输入框.name的值
// 也调用load 把filter过滤出的数组重新铺设
```